Parallélisme et Répartition-Correction

M1 IFI/MBDS, Décembre 2010, Session 1

2 heures, Feuille A4 manuscrite recto-verso autorisée

1 Recherche du minimum sur PRAM en temps poly-logarithmique

Le but de cet exercice est de proposer des algorithmes permettant de calculer le minimum d'une séquence d'entiers contenue dans un tableau $T[1,..,N]$ sur une machine PRAM.

1.1 Exécution en $O(1)$

Une première version de l'algorithme fonctionne de la façon suivante. Chaque processeur va comparer un élément du tableau initial ($T[i]$) avec un autre élément ($T[j]$ avec $i \neq j$)). Le résultat de cette comparaison sera mis dans un nouveau tableau. Dans une deuxième phase, l'algorithme utilisera ce tableau de résultats pour trouver le minimum.

1. Écrivez une boucle *for* permettant d'initialiser le nouveau tableau en $O(1)$. *Il y a deux façons de procéder, soit faire un tableau de taille N^2, soit faire un tableau de taille N. La version avec N est plus simple à écrire. La case result[i] va contenir le nombre de valeurs inférieures à $T[i]$.*

```
pour i de 1 à N en parallèle {
  result[i]=0;
}
```

2. Écrivez la boucle *for* qui effectue les comparaisons et remplis le nouveau tableau en $O(1)$.

```
pour i,j de 1 à N en parallèle {
   si (T[i]>T[j]) {
      result[i]=result[i]+1;
    }
}
```

3. Écrivez la boucle *for* qui affiche l'élément minimum du tableau en $O(1)$. *Par construction, si result[i] contient 0, c'est qu'aucun élément plus petit que $T[i]$ n'a été trouvé, c'est donc lui le plus petit*

```
pour i de 1 à N en parallèle {
  si(result[i]==0) { afficher(i); }
}
```

4. Combien de processeurs cet algorithme nécessite-t-il ? *Pour N éléments, on doit faire de l'ordre de N^2 comparaisons, donc $O(N^2)$ processeurs.*

5. Sur quel type de P-RAM peut-il être exécuté ? *On a des accés concurrents en lecture suivant les valeurs du couple (i, j), mais aussi des accés en écriture sur $result[i]$. Donc une CRCW.*

1.2 Réduction du nombre de processeurs

Nous allons essayer de réduire le nombre de processeurs nécessaires pour trouver le minimum. Soit A_1 l'algorithme écrit dans la section précédente. Soit A_2 l'algorithme fonctionnant de la façon suivante

- Le tableau initial est découpé (logiquement) en blocs de taille $\sqrt{n}$
- A_1 est appliqué sur chacun des blocs pour obtenir un tableau de $\sqrt{n}$ cases.
- A_1 est appliqué sur le tableau précédent

1. Exécutez A_2 sur le tableau $[4, 5, 17, 3, 8, 9, 0, 2, 20]$

2. Dans le cas général, que contient le tableau crée à l'etape 2 ? *Il contient le minimum de chaque bloc de taille $\sqrt{N}$.*

3. Montrez que A_2 nécessite $N^{1+\frac{1}{2}}$ processeurs *En partant d'un tableau de taille N, il va y avoir $\sqrt{N}$ blocs de $\sqrt{N}$ éléments. Or pour traiter un bloc, il faut $\sqrt{N} * \sqrt{N}$ comme montré dans l'exercice précédent. Nous avons donc besoin de $\sqrt{N} * \sqrt{N} * \sqrt{N} = N^{1+\frac{1}{2}}$*

4. Proposez un algorithme A_3 nécessitant $N^{1+\frac{1}{4}}$ processeurs. *On divise le tableau initial en blocs de taille $\sqrt{N}$ et on applique A_2 sur chacun d'eux. Donc on a $\sqrt{N}$ algorithmes A_2 en parallèle, chacun travaillant sur un tableau de taille $\sqrt{N}$. En utilisant le résultat de la question précédente, on trouve que chaque A_2 nécessite $N^{\frac{1}{2}+\frac{1}{4}}$ et donc un total de $\sqrt{N} * N^{\frac{1}{2}+\frac{1}{4}} = N^{1+\frac{1}{4}}$*

5. Décrire brièvement un algorithme A_k utilisant l'algorithme A_{k-1} pour trouver le minimum d'un tableau de N éléments (on ne demande pas d'écrire cet algorithme pour une PRAM).*On divise le tableau initial en blocs de taille $\sqrt{N}$ sur lesquels ont applique A_{k-1}*

6. Quel est le nombre de processeurs nécessaires pour A_k ? *En utilisant les résultats précédents, on voir que le nombre de processeurs est $N^{1+\frac{1}{2^{k-1}}}$. Il suffit de vérifier pour $A_1 = N^2$, $A_2 = N^{1+\frac{1}{2}}$....*

7. Quelle est la complexité en temps de A_k ? *La difficulté ici est de trouver le plus grand k pour une valeur de N donnée. Ça indique la profondeur de l'arbre et donc la complexité en temps de l'algorithme. Pour cela, remarquons que si A_k travaille sur un tableau de N cases, A_{k-1} travaille sur*

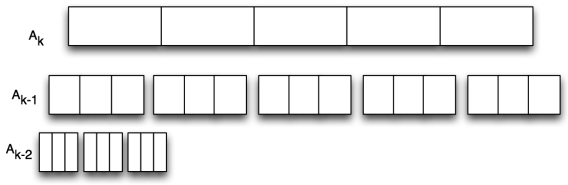

Figure 1: Représentation arborescente de l'algorithme

$\sqrt{N}$ tableaux de $\sqrt{N}$ cases, A_{k-2} sur des tableaux de $\sqrt[4]{N}$... Autrement dit, les blocs ont une taille de $N^{\frac{1}{2^i}}$ et l'algorithme s'arrête quand le bloc est trop petit pour être sub-divisé. Il suffit donc de résoudre $N^{\frac{1}{2^i}} = 2$, ce qui donne $i = O(log(log(N))$.

2 Tri fusion de Batcher sur PRAM

On rappelle que le tri fusion de Batcher utilise des comparateurs pour construire des réseaux $FUSION_m$, tel que $FUSION_1$ fusionne deux listes triées de 2^1 éléments.

1. Comment construit-on un réseau $FUSION_m$? Combien de comparateurs sont-ils nécessaires?
2. Combien faut-il de processeurs sur une PRAM pour implémenter un comparateur ?
3. En déduire le nombre de processeurs nécessaires pour implémenter $FUSION_m$
4. Comment construit-on TRI_m ?
5. Combien de processeurs sont-ils nécessaires pour implémenter TRI_m sur une PRAM ?

3 Élection de leader sur anneau avec passage de message

De nombreux algorithmes distribués nécessitent qu'un processus ait un rôle particulier et soit donc *leader*. Le but de cet exercice est d'écrire un algorithme en C/MPI pour élire un processus sur une topologie en anneau. Le *leader* sera le processus de plus haut rang actuellement sur l'anneau. L'algorithme est exécuté par le premier processus qui détecte l'absence de leader (suite à une panne par exemple).

- Un message *Election* contenant le rang du processus est envoyé au suivant sur l'anneau
- Chaque processus ajoute son rang dans le message et le transmet au suivant

- Quand le message revient au processus initial, il identifie le leader, et envoie sur l'anneau un message *OK* contenant le rang du leader

1. Combien de *Send* et de *Receive* sont effectués lors d'une élection sur un anneau de N processus?

2. Sachant que le temps de communication est de la forme $\beta + L.\tau$ et que le rang d'un processus occupe une taille de 1 dans un message, quelle est la durée totale d'une élection ?

3. Écrivez une méthode (en pseudo) C/MPI *election(int k)* permettant d'effectuer une élection. Cette méthode sera appelée par tous les processus en même temps mais seul le processus de rang k émettra le message.

4. On suppose qu'une panne arrive pendant une élection. Discutez des problèmes engendrés et des solutions possibles pour assurer l'élection lorsque c'est possible. On pourra considérer les différents cas suivant le processus qui tombe en panne (initiateur de l'élection ou autre) et suivant le moment de la panne (avant message *Election*, avant message *OK*...).

4 Anneau et Hypercube

On considère un anneau de 2^d noeuds et un hypercube de dimension d.

1. Combien de noeuds y'a-t-il dans un *d-cube* ?

2. Rappelez comment est construit un *d-cube* et comment son numérotés les sommets

3. Montrez qu'un anneau de 8 noeuds peut être construit sur un $3-cube$ (i.e. il est possible de relier les noeuds de l'hypercube sous forme d'un anneau).

4. Une idée pour le cas général de 2^d noeuds ?